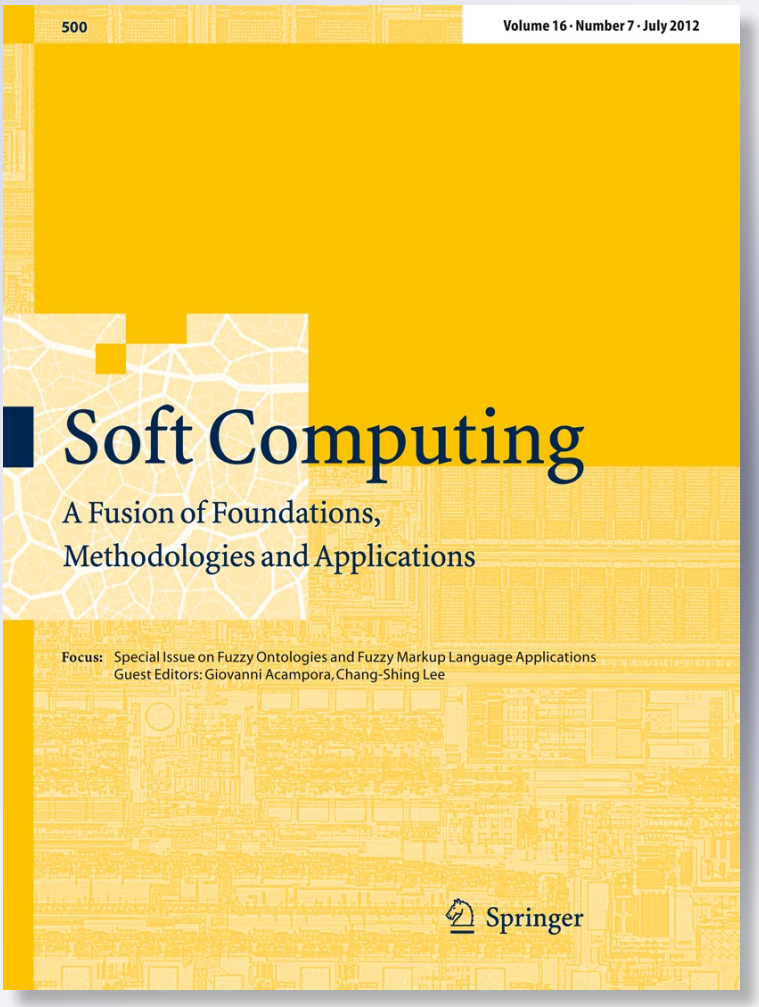
Particle swarm optimization with deliberate loss of information

C. A. Voglis, K. E. Parsopoulos & I. E. Lagaris

Soft Computing
A Fusion of Foundations, Methodologies and Applications

ISSN 1432-7643
Volume 16
Number 8

Soft Comput (2012) 16:1373-1392
DOI 10.1007/s00500-012-0841-5



Springer

Soft Comput (2012) 16:1373–1392
DOI 10.1007/s00500-012-0841-5

ORIGINAL PAPER

Particle swarm optimization with deliberate loss of information

C. A. Voglis · K. E. Parsopoulos · I. E. Lagaris

Published online: 28 March 2012

Abstract We introduce a new variant for the constriction coefficient model of the established particle swarm optimization (PSO) algorithm. The new variant stands between the synchronous and asynchronous version of PSO, combining their operation regarding the update and evaluation frequency of the particles. Yet, the proposed variant has a unique feature that distinguishes it from other approaches. Specifically, it allows the undisrupted move of all particles even though evaluating only a portion of them. Apparently, this implies a loss of information for PSO, but it also allows the full exploitation of the convergence dynamic of the constriction coefficient model. Moreover, it requires only minor modifications to the original PSO algorithm since it does not introduce complicated procedures. Experimental results on widely used benchmark problems as well as on problems drawn from real-life applications, reveal that the proposed approach is efficient and can be very competitive to other PSO variants as well as to more specialized approaches.

Keywords Particle swarm optimization · Asynchronous models · Loss of information

1 Introduction

Particle swarm optimization (PSO) is a population-based heuristic algorithm for numerical optimization. It was first introduced by Eberhart and Kennedy (1995) and Kennedy and Eberhart (1995) as an alternative to evolutionary algorithms (EAs) (Bäck et al. 1997) that were dominant at that time. PSO has many in common with EAs (Bäck et al. 1997). For instance, it uses a population of search points, new potential solutions are stochastically generated based on the information drawn from the best ones, and it requires only function values disregarding the existence of gradient information. These properties allow PSO to efficiently work on problems where only limited information on the objective function is available or accessible. Such problems are met in black-box optimization as well as in cases where computation is characterized by uncertainties, inaccurate data, noisy and time-dependent environments (Parsopoulos 2002, 2010).

Since its introduction, PSO has gained increasing popularity. This can be ascribed to its experimentally verified efficiency in challenging optimization problems as well as its easy implementation. The minor effort required to implement PSO in modern programming languages has attracted researchers from different scientific disciplines in search of a simple yet efficient optimization algorithm that can tackle complicated problems without strong mathematical prerequisites. As a result, there is a remarkable number of PSO-based applications in diverse scientific fields (Poli 2007), although several modifications in the original PSO model were necessary to enhance its performance on some occasions.

Clerc and Kennedy (2002) derived the stability analysis of PSO, which was later extended by Trelea (2003). In their analysis, several PSO models were considered and theoretically analyzed to distinguish the most promising one. These works revealed that a model with a constriction coefficient on the velocities, along with proper parameter configuration, could alleviate the deficiencies of the first PSO model. The parameter setup of this model was theoretically justified,

C. A. Voglis · K. E. Parsopoulos (✉) · I. E. Lagaris
Department of Computer Science, University of Ioannina, Ioannina, Greece
e-mail: kostasp@cs.uoi.gr

C. A. Voglis
e-mail: voglis@cs.uoi.gr

I. E. Lagaris
e-mail: lagaris@cs.uoi.gr

resulting in a PSO variant that proved to be popular especially among non-expert practitioners with limited knowledge of such algorithms, their operation and manipulation.

The theoretical investigation of the constriction coefficient model was mostly based on loose assumptions. However, due to the necessity for particular mathematical manipulations, the memory of the algorithm was unavoidably considered to remain fixed during its execution. In other words, the information that is used to attract the particles towards specific (promising) regions of the search space was not updated throughout the algorithm's run. Yet, in practice this memory is frequently updated. Hence, every change in memory can be considered as a restarting of the algorithm's dynamics. This observation has not attracted much attention although it can be highly related to PSO's efficiency. Indeed, it is easily verified that in a typical PSO run, even on a problem of moderate difficulty, the information stored in PSO's memory may continually change. Thus, the question naturally comes up: *how crucially is the performance of PSO affected by the continuously restarted dynamics?* Newer theoretical studies such as (Blackwell 2006) shed some light on this issue. Nevertheless, it still remains a rather neglected aspect of the PSO algorithm.

The present paper aims at experimentally probing this attribute of PSO. More specifically, we propose a constriction coefficient PSO model, called *Particle Swarm Optimization with Deliberate Loss of Information* (PSO-DLI), equipped with a stochastic decision-making scheme that determines whether a particle shall be evaluated and update its memory at each iteration. This scheme can be considered as a hybrid combination of the standard (synchronous) and the asynchronous PSO model, which is mostly used in parallel implementations. However, the proposed approach has a unique characteristic: every particle's motion is undisrupted even if it is not selected for evaluation and memory update. Thus, the algorithm can deploy its dynamic without continuous disruptions imposed by the dynamic's restarting described above.

On the other hand, it is easily understood that such a model suffers from *loss of information* at each iteration, namely unevaluated new particle positions. Thus, the current study has the mission to experimentally investigate the impact of such a loss of information on the algorithm's performance. For this purpose, experiments are conducted on a number of widely used benchmark problems as well as on problems drawn from real-life applications. More specifically, the standard test suite that consists of the Sphere, Rosenbrock, Rastrigin, Griewank and Ackley function, both in their original, generalized forms as well as in their rotated and shifted forms that were used for the CEC2008 competition on large-scale global optimization (Tang et al. 2007), were considered. Additionally, a set of nonlinear systems stemming from neurophysiology, chemical equilibrium, kinematic, combustion and economic modeling applications (Grosan 2008), was considered and solved with the proposed PSO-DLI approach. Finally, we considered a set of 19 benchmark problems that served as test suite for the recently published special issue on "*scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems*" of the present journal (Lozano et al. 2011). The special issue was devoted to empirical evaluations of several algorithms on large-scale instances of the specific problems.[1]

In our experiments, PSO-DLI was compared against the asynchronous PSO model on the standard test suite and, additionally, against genetic algorithms, differential evolution and the more sophisticated MONS approach (Grosan 2008) on the nonlinear systems. On the CEC'2008 test suite, it was compared also to more specialized and complicated algorithms such as EPUS-PSO and DMS-PSO (Hsieh et al. 2008; Zhao et al. 2008). The results revealed a remarkable potential of PSO-DLI to compete with all these algorithms on these problems. Finally, on the special issue's test suite, PSO-DLI was compared to the 16 algorithms that participated in the competition, exhibiting reasonably good performance. Considering that its implementation practically requires only marginal alteration of the existing PSO models and codes, we can suggest PSO-DLI as a very promising PSO variant that can be very useful especially to non-expert practitioners that wish to use a simple PSO model such as the constriction coefficient (or the algebraically equivalent inertia weight) one.

The rest of the paper is organized as follows: Sect. 2 offers the necessary background on PSO, both in its synchronous and asynchronous version. The proposed PSO-DLI algorithm is presented in Sect. 3 and experimental results are reported and discussed in Sect. 4. Finally, the paper concludes in Sect. 5.

2 Background information

For completeness purpose, a brief presentation of the required background information is given in the following sections. This includes the standard (synchronous) PSO algorithm as well as its asynchronous counterpart. These approaches will constitute the ground for the presentation of the proposed PSO-DLI scheme in a later section.

2.1 Particle swarm optimization

Consider the n-dimensional global optimization problem:

$$\min_{x \in X \subset \mathbb{R}^n} f(x).$$

PSO employs a set of potential solutions,

[1] Detailed descriptions are available at http://sci2s.ugr.es/EAMHCO.

$$S = \{x_1, x_2, \ldots, x_N\}, \quad x_i \in X, \quad i \in I = \{1, 2, \ldots, N\},$$

to probe the search space. This set is called a *swarm*, while each vector in S corresponds to a *particle*:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X, \qquad i \in I.$$

Each particle is randomly initialized in X and allowed to iteratively move within it. Its motion is determined by an adaptable *velocity*, while it retains in memory the *best position* it has ever visited. These quantities are denoted as:

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top, \quad p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top, \quad i \in I,$$

respectively. Thus, if t denotes the current iteration of the algorithm, it holds that:

$$p_i(t) = \arg \min_{q \in \{0,1,\ldots,t\}} \{f(x_i(q))\}, \quad i \in I.$$

Best positions constitute the *memory* of the particles and it is used to guide them towards the promising regions of the search space, i.e., regions that posses lower function values.

In order to avoid the premature convergence of the particles on local minimizers due to the sole use of own gathered information, the concept of *neighborhood* was introduced (Kennedy 1999; Suganthan 1999). Specifically, each particle assumes a set of neighboring particles with which it exchanges information, i.e., the best position among all the particles that constitute a neighborhood is communicated only among them and it is used for their velocity update. To put it formally, a neighborhood of the ith particle is defined as a set:

$$\begin{aligned} &\mathrm{NB}_{i,s} = \{j_1, j_2, \ldots, j_s\}, \quad j_k \in I, \\ &\quad k = 1, 2, \ldots, s, \quad i \in \mathrm{NB}_{i,s}, \end{aligned}$$

which consists of the indices of all the particles with which it can exchange information. Then, the neighborhood's best position:

$$p_{g_i} = \arg \min_{j \in \mathrm{NB}_{i,s}} \{f(p_j)\}, \tag{1}$$

is used along with p_i to update the ith particle's velocity at each iteration. The parameter s defines the number of particles that constitute the neighborhood and it is often called the *neighborhood size*. Obviously, it must hold that $1 \leq s \leq N$. By definition, the ith particle is included in its own neighborhood. In the special case where $s = N$, the whole swarm constitutes the neighborhood. The latter case defines the so-called *global* PSO model (denoted as *gbest*), while strictly smaller neighborhoods correspond to the *local* PSO model (denoted as *lbest*).

The schemes that are used for determining the particles that constitute each neighborhood are called *neighborhood topologies* and they have a crucial impact on PSO's performance. Probably, the most trivial choice is the random selection of s neighbors per particle. However, this scheme is accompanied with an undesirable lack of control of the information flow among the particles, which may prohibit the effective exploration of the search space within a reasonable number of iterations. Therefore, in practice it is rarely favored against alternative topologies, such as the *ring topology*. According to it, each particle assumes as neighbors the particles with its adjacent indices. The number of neighbors is determined by a parameter, r, called the *neighborhood radius*. Thus, a ring neighborhood of radius r of the ith particle, is defined as the set of indices:

$$\mathrm{NB}_{i,r} = \{i - r, i - r + 1, \ldots, i, \ldots, i + r - 1, i + r\},$$

where the indices recycle after the value N, i.e., the index 1 is assumed to follow immediately after N. The recycling gives the sense of moving on a clockwise ordered ring, justifying the name of this topology.

Algorithm 1: Pseudocode of the standard (synchronous) PSO algorithm.

Input : Objective function, $f : X \subset \mathbb{R}^n \to \mathbb{R}$; swarm size: N; parameters: χ, c_1, c_2.
Output: Best detected solution: x^*, $f(x^*)$.

// Initialization
$t \leftarrow 0$.
for $i \leftarrow 1$ **to** N **do**
  Initialize $x_i(t)$ and $v_i(t)$, randomly.
  $p_i(t) \leftarrow x_i(t)$. // Initialize best position
  Evaluate $f(x_i(t))$. // Evaluate particle
end
// Main Iteration
while *(termination criterion not met)* **do**
  // Position and Velocity Update
  for $i \leftarrow 1$ **to** N **do**
    Determine $p_{g_i}(t)$ from the i–th particle's neighborhood.
    for $j \leftarrow 1$ **to** n **do**
      $v_{ij}(t+1) \leftarrow \chi\left[v_{ij}(t) + c_1\mathcal{R}_1(p_{ij}(t) - x_{ij}(t)) + c_2\mathcal{R}_2(p_{g_i,j}(t) - x_{ij}(t))\right]$.
      $x_{ij}(t+1) \leftarrow x_{ij}(t) + v_{ij}(t+1)$.
    end
  end
  // Best Positions Update
  for $i \leftarrow 1$ **to** N **do**
    Evaluate $f(x_i(t+1))$. // Evaluate new position
    $p_i(t+1) \leftarrow \begin{cases} x_i(t+1), & \text{if } f(x_i(t+1)) < f(p_i(t)), \\ p_i(t), & \text{otherwise.} \end{cases}$
  end
  $t \leftarrow t + 1$.
end

Based on the definitions above, the iterative scheme of PSO is defined as follows (Clerc 2002):

$$\begin{aligned} v_{ij}(t+1) = \chi[&v_{ij}(t) + c_1\mathcal{R}_1(p_{ij}(t) - x_{ij}(t)) \\ &+ c_2\mathcal{R}_2(p_{g_i,j}(t) - x_{ij}(t))], \end{aligned} \tag{2}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{3}$$

where, $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, n$; the parameter χ is the *constriction coefficient*; acceleration constants c_1 and c_2 are called the *cognitive* and *social* parameter, respectively; and $\mathcal{R}_1, \mathcal{R}_2$, are random variables uniformly distributed in the range [0, 1]. It shall be noted that a different value of $\mathcal{R}_1$ and $\mathcal{R}_2$ is sampled for each i and j in Eq. (2). Also, the best position of each particle is updated at each iteration, as follows:

$$p_i(t+1) = \begin{cases} x_i(t+1), & \text{if } f(x_i(t+1)) < f(p_i(t)), \\ p_i(t), & \text{otherwise,} \end{cases} \quad i \in I. \tag{4}$$

The PSO variant described above was introduced by Clerc and Kennedy (2002) and it has gained increased popularity, especially in interdisciplinary applications. This comes on account of its close relevance to the original PSO model, which implies simplicity and efficiency, as well as on its theoretical support. Based on the stability analysis (Clerc 2002), the parameter set:

$$\chi = 0.729, \quad c_1 = c_2 = 2.05, \tag{5}$$

was determined as a satisfactory setting that produces a balanced convergence speed of the algorithm. Nevertheless, alternative settings have been introduced in relevant works (Trelea 2003).

The initialization of swarm and velocities is usually performed randomly and uniformly in the search space, although more sophisticated techniques can enhance the overall performance of PSO as reported in (Parsopoulos 2002a, b). Pseudocode of PSO is reported in Algorithm 1, where its essentially synchronous nature becomes apparent. By this, we mean that all particles update their velocities and current positions (lines 8–14 of Algorithm 1) prior to the update of their memory (lines 15–18). Alternative models that work rather asynchronously have also been proposed. Their basic elements are briefly presented in the next section.

2.2 Asynchronous particle swarm optimization

Asynchronous PSO models (henceforth denoted as PSO-ASY) have been developed as alternatives to the synchronous model described in the previous section. The main difference from the synchronous PSO is that, in a given iteration, each particle updates and communicates its memory to its neighbors immediately after its move to a new position. Thus, the particles that remain to be updated in the same iteration can exploit the new information immediately, instead of waiting for the next iteration as in the synchronous model.

Algorithm 2: Pseudocode of the parametric asynchronous PSO (PSO–ASY) algorithm. The value $\rho_{\text{asy}} = 0$ corresponds to the typical asynchronous PSO.

Input : Objective function, $f : X \subset \mathbb{R}^n \to \mathbb{R}$; swarm size: N; parameters: χ, c_1, c_2; probability of not updating: ρ_{asy}; random variable: $\mathcal{R} \sim U([0, 1])$.

Output: Best detected solution: x^*, $f(x^*)$.

```
// Initialization
1  t ← 0.
2  for i ← 1 to N do
3      Initialize x_i(t) and v_i(t).
4      Set p_i(t) ← x_i(t). // Initialize best position
5      Evaluate f(x_i(t)). // Evaluate particle
6  end
   // Main Iteration
7  while (termination criterion not met) do
       // Position, Velocity and Best Positions Update
8      for i ← 1 to N do
9          if (R > ρ_asy) then
10             Determine p_{g_i} from the i–th particle's neighborhood.
11             for j ← 1 to n do
12                 v_ij(t+1) ← χ[v_ij(t) + c1 R1 (p_ij(t) − x_ij(t)) + c2 R2 (p_{g_i,j} − x_ij(t))].
13                 x_ij(t+1) ← x_ij(t) + v_ij(t+1).
14             end
15             Evaluate f(x_i(t+1)). // Evaluate new position
16             p_i(t+1) ← { x_i(t+1), if f(x_i(t+1)) < f(p_i(t)),
                          { p_i(t),   otherwise.
17         end
18     end
19     t ← t + 1.
20 end
```

The differences from the synchronous PSO model become apparent in the pseudocode of Algorithm 2, where we can see the existence of a single for-loop (lines 8–18) on the number of particles. Thus, for the determination of p_{g_i} in line 10 of Algorithm 2, the latest findings of all particles in the neighborhood are taken into consideration even if they were achieved at the same iteration. This is the reason for dropping the iteration index of p_{g_i} in line 12, simply denoting p_{g_i} instead of $p_{g_i}(t)$. Contrary to this, the corresponding velocity update in line 11 of Algorithm 1 is solely based on the information coming from the previous iteration.

The impact of the asynchronous update in PSO's performance heavily depends on the given optimization problem and the considered implementation. In general, the asynchronous model is characterized by faster disclosure of new best positions than in the standard (synchronous) PSO model, thereby increasing convergence speed. This comes at the cost of getting trapped by rapidly attracting all particles to a deceitful solution. Thus, the preference between synchronous or asynchronous PSO shall be dictated by the specific problem at hand.

However, in practice it may also be dictated by the available hardware. For example, in parallel environments it is desirable to have the smallest possible idle time per processor. In such environments, the time required per function evaluation plays a significant role. If the

evaluations require almost identical time for any point in the search space, the synchronous PSO is satisfactory since all particles will be evaluated and updated almost concurrently. On the other hand, if there is significant time deviation for the evaluation of different points in the search space, the asynchronous model is expected to outperform the synchronous one with respect to the required running time.

Actually, Algorithm 2 introduces a parametric PSO-ASY model. The parameter ρ_{asy} is used to simulate both the serial (single-processor) and parallel implementation of PSO-ASY by properly tuning its value in line 9, as follows:

$$\begin{aligned} &\rho_{\text{asy}} = 0 \,(\text{serial PSO-ASY}),\\ &0 < \rho_{\text{asy}} < 1 \,(\text{parallel PSO-ASY}). \end{aligned} \tag{6}$$

Several asynchronous PSO variants have been proposed in the literature (Akat 2008; Desell et al. 2009; Gaz 2007; Hernane et al. 2010; Koh et al. 2006). Most of these approaches refer to parallel implementations of PSO that were shown to be competitive or even superior to the synchronous ones. Nevertheless, even in serial implementations the asynchronous models have exhibited promising results.

In the next section, the proposed PSO-DLI algorithm is described.

3 Particle swarm optimization with deliberate loss of information

The inspiration behind the development of the proposed *Particle Swarm Optimization with Deliberate Loss of Information* (PSO-DLI) algorithm, sprang from the analysis of Clerc and Kennedy (2002) on the dynamics of the PSO variant defined by Eqs. (2) and (3) in Sect. 2.1. Based on this analysis, the constriction coefficient model of PSO was distinguished among others on the basis of its good convergence properties. The analysis was based on the assumption that the best positions of the particles remain fixed throughout the algorithm's execution, aiming at the determination of proper model and parameter configurations such that the swarm consistently converges on sub-optimal solutions. In fact, Clerc and Kennedy proved convergence towards a state where the velocities vanish, while the particles' positions approximate a linear combination of the two (fixed) best positions involved in Eq. (2).

However, in realistic cases the best positions of the particles do not remain fixed. Even in problems of moderate difficulty, it is experimentally verified that the best positions are frequently changing. This implies that the dynamics of the model are restarted after each best position update, prohibiting the particles from taking full advantage of the model's convergence properties. Moreover, it has been experimentally observed that, especially at the later stages of the optimization procedure, only some of the best positions' updates offer critical new information to the particles, while the rest lead only to marginal improvements.

These critical observations suggest that a particle may not be able to deploy its exploration/exploitation capability due to frequent changes of its best position, which may be rather unnecessary with respect to the gained improvement. In addition, each best position update implies that a function evaluation preceded. It is widely accepted that the total number of function evaluations constitutes the main computational cost of optimization algorithms. This is based on the reasonable assumption that, in difficult problems, the time required for the evaluation of the underlying objective function dominates the rest of the algorithm's operations. Thus, the regular evaluation of each particle and its best position update at every single iteration of the algorithm, may not only disrupt its convergence dynamics but also aggravates it with possibly futile additional cost whenever its discoveries are of minor importance.

Algorithm 3: Pseudocode of the PSO–DLI algorithm.

Input : Objective function, $f : S \subset \mathbb{R}^n \to \mathbb{R}$; swarm size: N; parameters: χ, c_1, c_2; loss probability: ρ_{loss}; random variable: $\mathcal{R} \sim U([0,1])$.
Output: Best detected solution: x^*, $f(x^*)$.

```
// Initialization
1  t ← 0.
2  for i ← 1 to N do
3  |  Initialize x_i(t) and v_i(t).
4  |  p_i(t) ← x_i(t). // Initialize best position
5  |  Evaluate f(x_i(t)). // Evaluate particle
6  end
   // Main Iteration
7  while (termination criterion not met) do
   |  // Position and Velocity Update
8  |  for i ← 1 to N do
9  |  |  Determine p_{g_i} from the i–th particle's neighborhood.
10 |  |  for j ← 1 to n do
11 |  |  |  v_ij(t+1) ← χ [v_ij(t) + c1 R1 (p_ij(t) − x_ij(t)) + c2 R2 (p_{g_i,j} − x_ij(t))].
12 |  |  |  x_ij(t+1) ← x_ij(t) + v_ij(t+1).
13 |  |  end
14 |  end
   |  // Evaluation and Best Position Update
15 |  for i ← 1 to N do
16 |  |  if (R > ρ_loss) then
17 |  |  |  Evaluate f(x_i(t+1)). // Evaluate new position
18 |  |  |  p_i(t+1) ← { x_i(t+1), if f(x_i(t+1)) < f(p_i(t)),
   |  |  |             { p_i(t),   otherwise.
19 |  |  end
20 |  end
21 |  t ← t + 1.
22 end
```

The main idea behind PSO-DLI is the alleviation of consistent evaluation and best position update of each particle at each iteration, without disrupting the particles' motion. This can be achieved by allowing each particle to move to a new position using Eqs. (2) and (3) but probabilistically deciding whether it shall be evaluated and

update its best position or not. If the particle is eventually evaluated then its best position update follows the standard procedure of Eq. (4). On the other hand, if it is not selected for evaluation then it simply remains to the new current position, ignoring its function value and retaining its previous best position.

By accepting this loss of information, i.e., the neglected function value at the new position, the available computational budget (maximum number of function evaluations) of the algorithm is retained, without interrupting the particle's convergent motion towards the already detected best positions. Naturally, this fosters the danger of possibly disregarding significant information. Nevertheless, the proposed scheme is expected to be advantageous under the (experimentally verified) assumption that a large number of best position updates do not offer radical information, i.e., they do not enhance the particle's exploration capability. In fact, they suppress its exploitation activity stemming from its theoretically implied tendency for convergence towards an aggregation of previously detected best positions.

A simple scheme for probabilistically selecting the particles that will loose information, can be based on a fixed user-defined probability value, $\rho_{\text{loss}} \in [0, 1]$. Then, for the ith particle at iteration t, a uniformly distributed random number, $\mathcal{R} \in [0, 1]$, is sampled and the particle is evaluated and updates its best position only if:

$$\mathcal{R} > \rho_{\text{loss}}. \tag{7}$$

Otherwise, it retains its best position and suffers the loss of its current position's function value. At a first glance, the proposed scheme seems to simply resemble the procedure of the asynchronous PSO model described in Sect. 2.2. This is partially true, since the evaluation procedure of PSO-DLI is asynchronous in nature. However, it has the unique characteristic that the particles that suffer loss of information are not held at their previous positions but they are allowed to follow the dynamics of the constriction coefficient model by moving to a new (unevaluated) current position. This way, it is expected that the gain from the particle's exploitation capabilities, along with its convergent behavior, will offer significantly better information when it will be eventually evaluated at a later iteration. To the best of the authors' knowledge, this property has not been considered in the established asynchronous models up-to-date.

The PSO-DLI scheme is clarified in the pseudocode of Algorithm 3. The main difference from Algorithms 1 and 2, lies beneath line 15 of Algorithm 3 where the randomized evaluation and update decision takes place. Notice that the particle moves without intervention to the new position (lines 11–12) regardless of the decision taken for its evaluation and best position update. Obviously, since PSO-DLI is expected to perform less function evaluations per iteration than the original PSO, the total number of iterations that can be conducted for a specific budget of $F_{\max}$ function evaluations is higher for PSO-DLI than the standard synchronous PSO.

Also, the implementation of PSO-DLI requires only minor additional effort than PSO, since there are no complicated procedures incorporated into the algorithm. Thus, it is easily understood that it can be very useful in applications where the original constriction coefficient (or its equivalent inertia weight) PSO model is frequently used. Moreover, PSO-DLI retains all the advantages of the PSO scheme regarding its potential for parallelization and combination with other global or local optimization algorithms. Additionally, this is accompanied by a remarkable increase in efficiency as it is shown in the next section.

4 Experimental results

PSO-DLI constitutes a minor modification of the standard constriction coefficient PSO model, although with a great performance impact. It aims at offering an efficient alternative to the standard PSO model in relative applications. Thus, the experimental setting in the present work primarily targeted at a first validation of PSO-DLI in widely used test problems and challenging applications, against the specific PSO model as well as standard versions of different popular EAs for tackling such problems.

For this purpose, four rounds of experiments were conducted. In the first round, PSO-DLI was validated against the standard synchronous and asynchronous PSO model on five widely used benchmark problems, providing the first evidence on its efficiency. In the second round of experiments, the same algorithms were applied on six real-world application problems modeled as systems of nonlinear equations. Additionally to the aforementioned comparisons, PSO-DLI was also compared against different evolutionary approaches on these problems. For this purpose, a Genetic Algorithm (GA), the basic Differential Evolution (DE) variant as well as the MONS approach (Grosan 2008) was considered.

In the third round of experiments, PSO-DLI was further assessed against established EAs on harder instances of the five benchmark problems used in the first round. More specifically, rotated and shifted versions of these problems, which have been used as benchmarks for the IEEE CEC'08 large-scale optimization competition (Tang et al. 2007), were considered. In this framework, we probed PSO-DLI's competitiveness against two specialized PSO-based approaches for large-scale optimization, namely the Efficient Population Utilization Strategy for Particle Swarm Optimizer (EPUS-PSO) (Hsieh et al. 2008, 2009) and the Dynamic Multi-Swarm Particle Swarm Optimizer with local search (DMS-PSO) (Zhao et al. 2008).

In the last round of experiments and despite the fact that PSO-DLI was neither designed nor tuned for large-scale problems, we took a further step and applied it on the test suite used in a recently published special issue of Soft Computing (Lozano et al. 2011). Our main consideration in this case was the evaluation of its ability to perform closely to the 16 algorithms presented in the special issue.

In all experiments, the employed PSO-DLI was based on the lbest variant of the constriction coefficient PSO model with a ring topology of radius 1. This choice was driven mostly by its popularity and good exploration properties. Regarding its parameters, the default values, $\chi = 0.729$, $c_1 = c_2 = 2.05$, defined in Eq. (5) were used. All the considered test problems are described in Appendix with the exception of the 19 problems of the fourth round of experiments that were omitted due to space limitations. However, the interested reader can have full access in all relevant information and sources through the web site http://sci2s.ugr.es/EAMHCO. In all experiments, the swarm was randomly and uniformly initialized within the corresponding search space.

4.1 First round of experiments: standard test suite

In the first round of experiments, the performance of PSO-DLI was assessed on the five widely used benchmark problems defined in "Standard test suite" of Appendix. These problems constitute the *standard test suite* and will be henceforth denoted as TP0–TP4. Their popularity is attributed to their different features, including unimodality/multimodality, separability/non-separability, generalization in arbitrary dimension as well as strongly/loosely correlated variables. The dimensions and range considered for each problem of the standard test suite in our experiments are reported in Table 1.

A number of comparative experiments were conducted on the standard test suite. More specifically, we employed the synchronous lbest PSO model, the corresponding asynchronous PSO-ASY model as well as the proposed PSO-DLI algorithm. The PSO-ASY model was considered with probabilities:

$$\rho_{\text{asy}} \in \{0.0, 0.3, 0.6, 0.9\},$$

which correspond to both its serial and parallel variant. The corresponding variants will be henceforth denoted as ASY0.0, ASY0.3, ASY0.6 and ASY0.9, respectively. PSO-DLI was considered with the same loss probabilities:

$$\rho_{\text{loss}} \in \{0.0, 0.3, 0.6, 0.9\}.$$

Notice that $\rho_{\text{loss}} = 0.0$ corresponds to the standard PSO algorithm (no information loss). The rest will be henceforth denoted as DLI0.3, DLI0.6 and DLI0.9, respectively.

For each problem instance and algorithm variant, 100 independent experiments were conducted. At each experiment, the algorithms were allowed to perform a total number of $F_{\max} = 1000 \times n$ function evaluations using a swarm of $N = 10 \times n$ particles, where n is the dimension of the corresponding problem instance. The employed parameter values are summarized in Table 2.

At the end of each experiment, the following quantities were recorded:

(a) the best detected solution and its function value;
(b) the number of iterations performed until $F_{\max}$ was reached;
(c) the total number of best positions updates, i.e., the total number of times where the particles discovered better best positions than the ones they had possessed.

Obviously, for the standard synchronous PSO model the number of iterations referred in (b) was always equal to $F_{\max}/N$. This is a consequence of the fixed number of function evaluations performed per iteration in standard PSO. However, this is not always the case for PSO-ASY and PSO-DLI since some particles may skip evaluation. Thus, these approaches are expected to perform more iterations than PSO for a given number of function evaluations. Also, the recorded quantity in (c) was considered as a measure of the swarm's exploitation activity.

The results are numerically reported in Table 3. More specifically, for each problem instance, the mean and the standard error (sample's standard deviation) of the best attained solution values are reported, along with the

Table 1 Dimension and range considered for each problem of the standard test suite

Problem	Dimension (n)	Range
TP0	10, 50, 100	$[-100, 100]^n$
TP1	10, 50, 100	$[-30, 30]^n$
TP2	10, 50, 100	$[-5.12, 5.12]^n$
TP3	10, 50, 100	$[-600, 600]^n$
TP4	10, 50, 100	$[-20, 30]^n$

Table 2 Parameter values for the considered PSO, PSO-ASY and PSO-DLI algorithms

Parameter	Value
PSO parameters	$\chi = 0.729$, $c_1 = c_2 = 2.05$
Neighborhood topology	Ring
Neighborhood radius	1
ρ_{asy}	0.0, 0.3, 0.6, 0.9
ρ_{loss}	0.0, 0.3, 0.6, 0.9
Problem dimensions	$n = 10, 50, 100$
Function evaluations	$F_{\max} = 1000 \times n$
Swarm size	$N = 10 \times n$

Table 3 Results for the standard test suite

Prob.	Dim.	Alg.	Best function value		Iterations		Best pos. updates	
			Mean	St. err.	Mean	St. err.	Mean	St. err.
TP0	10	PSO	3.6083e+00	2.0382e+00	100.00	0.00	2852.52	65.80
		ASY0.9	2.0666e+00	1.0908e+00	990.73	9.15	2923.25	62.55
		DLI0.9	**1.8243e−01**	**1.1637e−01**	990.50	9.81	5073.08	56.84
	50	PSO	8.8099e+03	9.5958e+02	100.00	0.00	12340.75	111.06
		ASY0.0	7.1619e+03	7.7548e+02	100.00	0.00	11817.68	141.32
		DLI0.9	**4.9309e+03**	**6.7732e+02**	991.46	4.10	21468.76	155.98
	100	PSO	4.8083e+04	2.9125e+03	100.00	0.00	21907.76	168.02
		ASY0.0	3.8759e+04	2.4937e+03	100.00	0.00	20452.06	189.29
		DLI0.9	**3.6088e+04**	**2.3608e+03**	991.73	3.02	40540.21	190.18
TP1	10	PSO	2.3688e+03	1.7897e+03	100.00	0.00	2879.06	73.15
		ASY0.0	1.2695e+03	8.7053e+02	100.00	0.00	2801.65	75.80
		DLI0.9	**3.7761e+02**	**3.0100e+02**	991.82	8.55	4836.70	77.18
	50	PSO	7.3816e+08	1.5687e+08	100.00	0.00	11947.84	141.34
		ASY0.0	5.1863e+08	1.2137e+08	100.00	0.00	11382.51	157.89
		DLI0.9	**4.2367e+08**	**8.7230e+07**	991.23	5.05	20391.61	144.05
	100	PSO	7.7595e+09	1.1348e+09	100.00	0.00	20954.89	190.16
		ASY0.0	**5.5733e+09**	**7.7423e+08**	100.00	0.00	19551.07	200.96
		DLI0.6	6.3030e+09	8.3405e+08	248.82	0.69	30397.02	200.18
TP2	10	PSO	1.5974e+01	3.7729e+00	100.00	0.00	1606.52	103.41
		ASY0.6	1.5632e+01	3.9773e+00	248.96	1.84	1634.77	95.19
		DLI0.9	**1.2054e+01**	**3.5569e+00**	991.67	9.61	3303.87	133.73
	50	PSO	3.5077e+02	2.0980e+01	100.00	0.00	7249.33	166.52
		ASY0.0	3.3301e+02	1.7513e+01	100.00	0.00	7170.33	206.54
		DLI0.9	**2.9829e+02**	**2.0173e+01**	991.93	4.22	14960.16	244.08
	100	PSO	9.2889e+02	3.0456e+01	100.00	0.00	13745.24	204.94
		ASY0.0	8.8769e+02	3.1189e+01	100.00	0.00	13349.63	235.92
		DLI0.9	**8.4038e+02**	**2.7152e+01**	991.22	2.88	28936.89	304.30
TP3	10	PSO	8.5357e−01	1.1730e−01	100.00	0.00	2757.86	74.05
		ASY0.9	7.3688e−01	1.5981e−01	989.61	9.99	2777.12	74.67
		DLI0.9	**3.9775e−01**	**1.1857e−01**	990.94	10.37	4570.84	91.24
	50	PSO	8.0954e+01	9.0157e+00	100.00	0.00	12370.88	122.43
		ASY0.0	6.4251e+01	7.9251e+00	100.00	0.00	11841.51	135.51
		DLI0.9	**4.4798e+01**	**5.7317e+00**	991.63	4.63	21478.51	122.69
	100	PSO	4.3306e+02	2.5046e+01	100.00	0.00	21891.44	167.52
		ASY0.0	3.5199e+02	2.3117e+01	100.00	0.00	20436.88	193.86
		DLI0.9	**3.2245e+02**	**1.9926e+01**	991.02	3.56	40528.91	213.32
TP4	10	PSO	2.0594e+00	4.4949e−01	100.00	0.00	2481.98	59.29
		ASY0.0	1.7063e+00	5.1973e−01	100.00	0.00	2459.59	58.48
		DLI0.9	**5.5583e−01**	**3.6882e−01**	991.00	9.92	4370.18	79.85
	50	PSO	1.3696e+01	4.0415e−01	100.00	0.00	10408.97	113.09
		ASY0.0	1.2839e+01	4.0867e−01	100.00	0.00	10159.90	106.90
		DLI0.9	**1.2168e+01**	**4.3272e−01**	991.50	4.03	17455.21	161.02
	100	PSO	1.7297e+01	2.3998e−01	100.00	0.00	17190.70	182.37
		ASY0.0	**1.6362e+01**	**2.4880e−01**	100.00	0.00	16611.49	161.07
		DLI0.9	1.6582e+01	3.4494e−01	991.46	3.12	30724.13	239.18

The best-performing PSO-ASY and PSO-DLI variants are reported. The best mean value per dimension appears boldfaced

corresponding numbers of iterations and best position updates (all averaged over 100 experiments) for the standard PSO as well as for the *best-performing* PSO-ASY and PSO-DLI variants. The smallest mean function value per problem instance is boldfaced along with its standard error.

The results are also graphically illustrated in Fig. 1. Specifically, the distributions of the attained best solution values are depicted in boxplots per problem instance and algorithm. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually (red crosses). Moreover, the notches define 95 % confidence intervals of median equality for box-to-box comparisons.

Each row of boxplots in Fig. 1 corresponds to a specific problem dimension (10, 50 or 100) and it is followed by a corresponding row of bargraphs that demonstrate the *percentage (%) of performance improvement* achieved by PSO-ASY and PSO-DLI against PSO, with respect to the mean (black bars) and standard error (white bars) of the best attained solution values. Obviously, negative bars indicate percentage of performance decline. These graphs aim at providing visual comparisons between the algorithms, using the standard PSO as reference point. A more compact graphical representation of the relative performance of PSO-DLI and PSO-ASY with respect to PSO, is offered in the left part of Fig. 3.

The derivation of sound conclusions regarding the performance differences between the algorithms was further facilitated by statistical significance tests. For this purpose, the nonparametric Wilcoxon rank-sum test was used for pairwise comparisons of PSO-DLI with PSO-ASY and PSO under the null hypothesis that "*the best solution values achieved by the compared algorithms are drawn from identical continuous distributions with equal medians*" at a 95 % level of significance. Then, the total number of cases (out of 15) where PSO-DLI had statistically significant better performance than the other approaches was recorded and it is reported in Fig. 2i. Finally, the frequency of appearance of each variant of PSO-ASY and PSO-DLI as the best-performing among all variants of the same algorithm per problem dimension is reported in Fig. 2ii and iii, respectively.

A first inspection of the results presented in Table 3 and Fig. 1 offers some interesting conclusions. As we can see, both PSO-ASY and PSO-DLI clearly outperformed the standard PSO in all cases. PSO-DLI achieved the overall best performance in all but two cases, namely the 100-dimensional instances of TP1 and TP4, where it was highly competitive to the dominant PSO-ASY variant. This is clearly derived also from Figs. 2i and 3. Moreover, DLI0.9 was the most promising PSO-DLI variant in almost all problem instances, with a single exception in TP1. This is depicted also in Fig. 2iii. The superiority of DLI0.9 with respect to the mean attained best solution value was habitually accompanied by the smallest standard errors, which is indicative of its robustness. On the other hand, ASY0.0 was the dominant among the PSO-ASY variants especially for higher-dimensional cases as we can see in Fig. 2ii. However, in the 10-dimensional cases the other PSO-ASY variants were also distinguished. This indicates sensitivity of PSO-ASY on the problem's dimension and structure.

A closer inspection of the results enriches our knowledge on PSO-DLI's performance. The last two columns of Table 3 show that PSO-DLI always performs the largest number of best position updates. This is a sound indication of PSO-DLI's better exploitation activity that is evidently promoted by the undisrupted motion of the particles. This evidence, combined with the dominance of DLI0.9 against the other PSO-DLI variants, suggests that higher values of the loss probability can be associated with better performance. As expected, the higher number of best position updates is associated with a higher mean number of iterations as it is displayed in the corresponding columns of Table 3. This is a natural consequence of PSO-DLI's probabilistic particle-update scheme as it was explained in Sect. 3.

Regarding the effect of problem's dimension on performance, Fig. 1 draws a clear picture. Observing the provided bargraphs in columns (same problem, increasing dimension), we can see an undisputed decline of the attained improvement (recall that bargraphs denote percentage of improvement/decline). This can be attributed to the total number of function evaluations, $F_{\max}$, which was linearly increased with the dimension (recall that $F_{\max} = 1000 \times n$), while there are experimental evidence that the degree of difficulty of a given problem increases exponentially with its dimension.

Additionally, the verified influence of the swarm size on PSO's dynamic (Bartz-Beielstein et al. 2004) impelled us to consider the case of swarm size scaling in order to verify the potential superiority of PSO-DLI against PSO. For this purpose, the experiments for PSO-DLI and PSO were repeated for all problems and dimensions, with different swarm sizes, $N = 1 \times n, 2 \times n, \ldots, 10 \times n$. Statistical significance tests were conducted between the algorithms to reveal the dominant approach. The results are reported in the right part of Fig. 3. More specifically, for each problem and dimension, a bar illustrates the number of different (out of 10) swarm sizes where the one algorithm outperformed (with statistical significance) the other as well as the number of cases with statistically indistinguishable performance. As we can see, the results are in good agreement with those previously presented for higher swarm size, with PSO being competitive only on a few cases, mostly for lower dimensions.

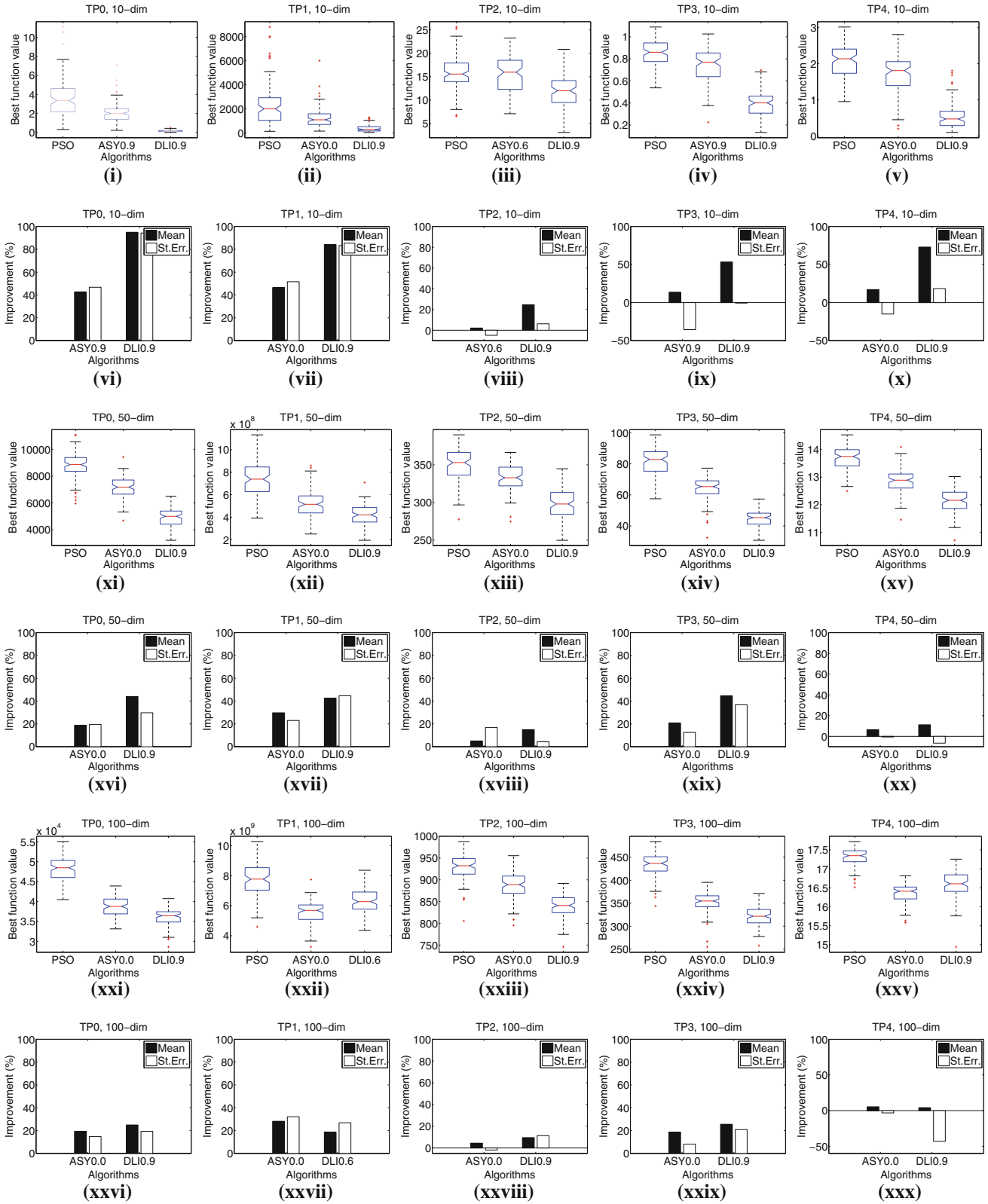


Fig. 1 Graphically illustrated results for the standard test suite

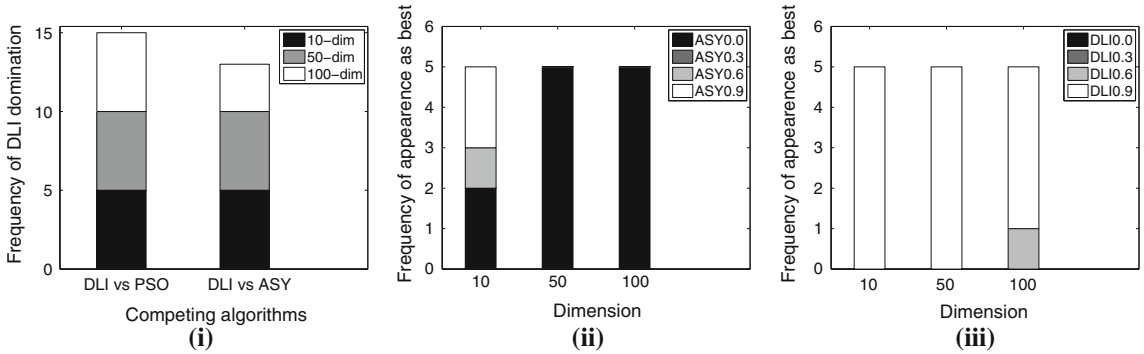


Fig. 2 Frequencies of domination among different algorithm variants

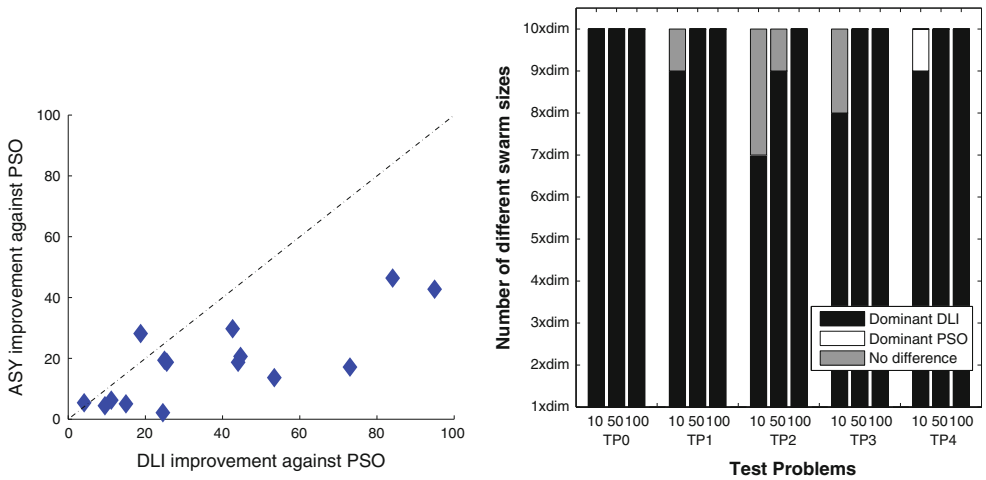


Fig. 3 *Left* relative performance of PSO-DLI and PSO-ASY with respect to the standard PSO. *Right* domination scaling between PSO-DLI and PSO for different swarm sizes

In conclusion, the experiments on the standard test suite suggest that PSO-DLI can be very competitive to fundamental synchronous and asynchronous PSO schemes. In order to attain further intuition on PSO-DLI's performance on less typical benchmark problems, further experiments were conducted on a set of challenging applications that involve the solution of nonlinear systems. The obtained results are reported in the next section.

4.2 Second round of experiments: nonlinear systems

In the second round of experiments, the real-world applications defined in "Nonlinear systems" of Appendix were considered. The corresponding problems are modeled as systems of nonlinear equations and they are formulated as global optimization problems through a transformation described in "Nonlinear systems". These problems will be henceforth denoted as TP5–TP10 and their dimensions and ranges are reported in Table 4. The same variants of the algorithms as in the previous section were considered, along with the parameter values reported in Table 2 (excluding problem dimensions). The presentation of the results follows closely the one in the previous section. Thus, data are numerically reported in Table 5 and graphically illustrated in Figs. 4 and 5.

The general picture of the results is aligned to that of the previous section. Namely, PSO-DLI outperformed PSO-ASY and PSO in all test problems, offering significant solution improvements that range from almost 30 % up to more than 90 % as it is depicted in the bargraphs of Fig. 4. This superiority was also supported by the statistical tests

Table 4 Dimension and range for the nonlinear systems

Problem	Dim. (n)	Range
TP5	10	$[-2, 2]^{10}$
TP6	6	$[-10, 10]^{6}$
TP7	5	$[-10, 10]^{5}$
TP8	8	$[-10, 10]^{8}$
TP9	10	$[-10, 10]^{10}$
TP10	20	$[-10, 10]^{20}$

Table 5 Results for the nonlinear systems

Prob.	Dim.	Alg.	Best function value		Iterations		Best pos. updates	
			Mean	St. err.	Mean	St. err.	Mean	St. err.
TP5	10	PSO	6.9206e−02	1.7539e−02	100.00	0.00	2643.84	64.31
		ASY0.6	6.2136e−02	1.7546e−02	249.00	2.16	2655.26	55.02
		DLI0.9	**7.3004e−03**	**2.8412e−03**	991.20	7.78	4976.99	61.87
TP6	6	PSO	2.7646e−02	2.4823e−02	100.00	0.00	1151.45	55.97
		ASY0.0	2.0806e−02	1.3875e−02	100.00	0.00	1127.38	52.72
		DLI0.9	**4.2369e−03**	**7.5450e−03**	992.15	11.38	2405.93	77.88
TP7	5	PSO	2.6403e−01	1.2727e−01	100.00	0.00	1018.52	49.26
		ASY0.0	2.1918e−01	1.2151e−01	100.00	0.00	998.23	48.10
		DLI0.6	**1.7708e−01**	**1.0610e−01**	249.05	2.77	1513.34	64.95
TP8	8	PSO	6.1197e−01	2.0500e−01	100.00	0.00	1783.93	74.10
		ASY0.9	5.3958e−01	1.9739e−01	990.99	11.44	1800.48	65.31
		DLI0.6	**4.2925e−01**	**2.0520e−01**	249.05	2.33	2556.96	85.26
TP9	10	PSO	2.9799e−01	1.5649e−01	100.00	0.00	2248.40	76.63
		ASY0.0	2.3905e−01	1.3167e−01	100.00	0.00	2198.38	81.57
		DLI0.6	**1.7411e−01**	**9.0344e−02**	248.84	2.18	3102.72	96.39
TP10	20	PSO	4.6172e−03	4.5435e−03	100.00	0.00	3016.80	78.69
		ASY0.6	3.3771e−03	2.5183e−03	249.11	1.46	3010.53	99.24
		DLI0.6	**1.6098e−04**	**1.9826e−04**	249.15	1.46	4232.29	108.90

The best-performing PSO-ASY and PSO-DLI variants are reported. The best mean value per dimension is boldfaced

and it is reflected in PSO-DLI's domination depicted in Fig. 5i. Moreover, the relative improvements achieved by PSO-DLI and PSO-ASY against PSO were always in favor of the proposed approach as illustrated in Fig. 5iii. Indeed, as we can see PSO-ASY's improvements were limited in a range between 10 % and slightly over 20 %, while at the same time PSO-DLI's improvements soared over 80 %. The two distinguishable clusters in the scatter plot of Fig. 5iii may be the results of possible structural resemblance of the corresponding problems, although this speculation requires further investigation that lies out of the scope of this paper.

An interesting evidence from this round of experiments was the designation of DLI0.6 as the best among PSO-DLI variants in more than half of the problems. This is in agreement with the expectation of better PSO-DLI performance under higher loss-probability values, which was identified in the standard test suite. However, it can be conceived also as an evidence of a plausible dependence of the most efficient variant on the corresponding problem at hand. Nevertheless, either PSO-DLI variants retained the highest number of best positions updates in all cases.

The second round of experiments was enriched with further comparisons of PSO-DLI with other established algorithms. More specifically, a steady state Genetic Algorithm (GA) (Goldber 1989) as well as the Differential Evolution (DE) (Storn 1997) algorithm were tested against different PSO-DLI variants under the configuration reported in Table 6. The employed GA approach was implemented using the GAlib[2] software. In order to facilitate comparisons also with the results provided in Grosan (2008) for the multi-objective MONS approach, the experimental setup reported in Table 7 was adopted for all algorithms.

In accordance to the previous experiments, we performed 100 independent runs per algorithm using the provided maximum number of function evaluations and recording the best solution value attained per experiment. The obtained results are reported in Table 8 with respect to the mean and standard error of the best function values. The results of the MONS approach were adopted from the original source (Grosan 2008) where no standard errors are reported.

Some interesting observations can be made in Table 8. Firstly, we can observe that all PSO-DLI variants were among the best-performing algorithms in all problems. Particularly in TP5, TP6 and TP10, they outperformed the rest of the algorithms. However, DE was shown to outperform PSO-DLI in the rest of the test problems. Also, contrary to our previous findings, DLI0.3 was shown to be superior than DLI0.6 and DLI0.9.

A reasonable explanation for this outcome lies in the specific experimental setup. More specifically, the number of function evaluations adopted from (Grosan 2008) was

[2] http://lancet.mit.edu/galib-2.4.

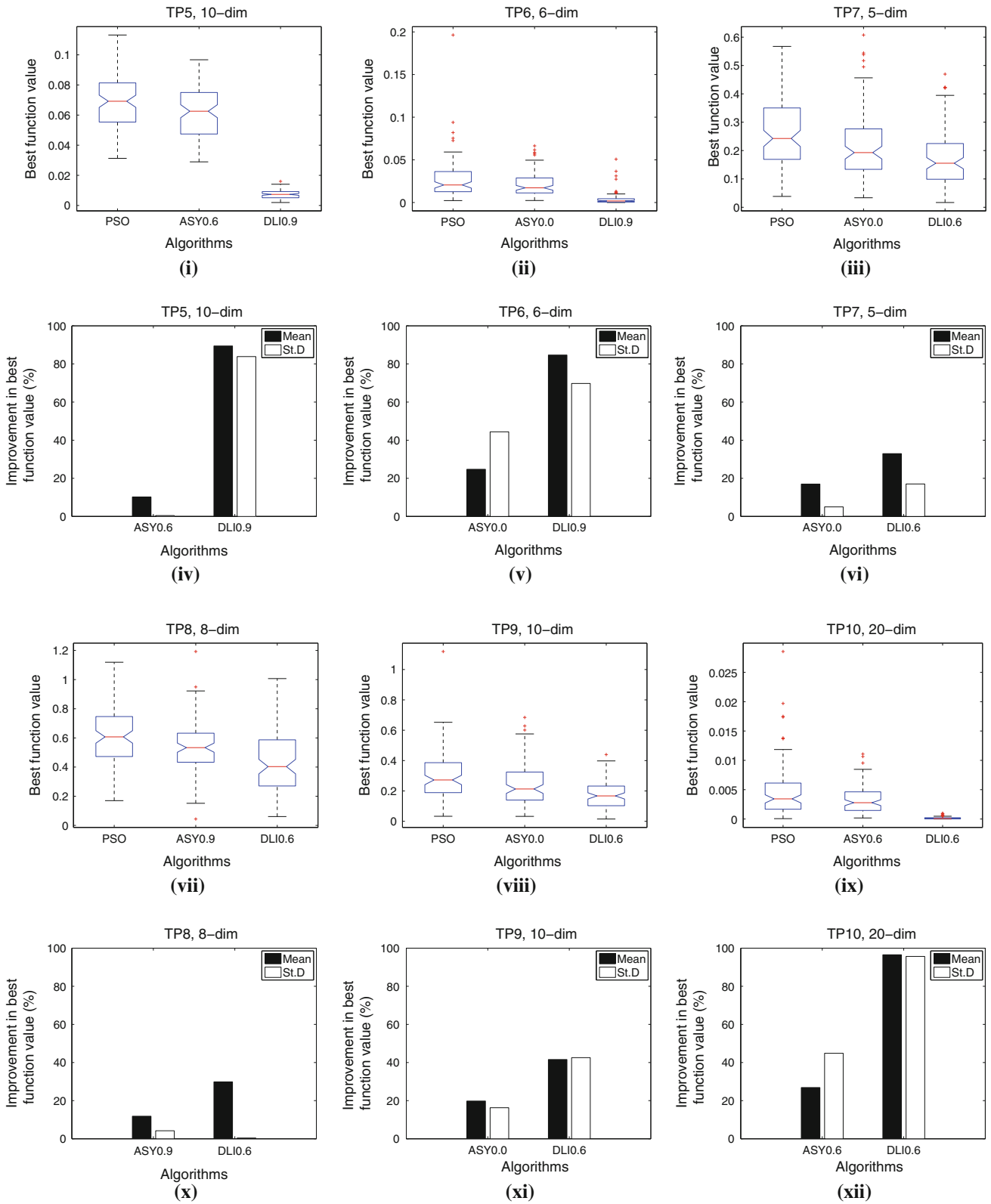


Fig. 4 Graphically illustrated results for the nonlinear systems

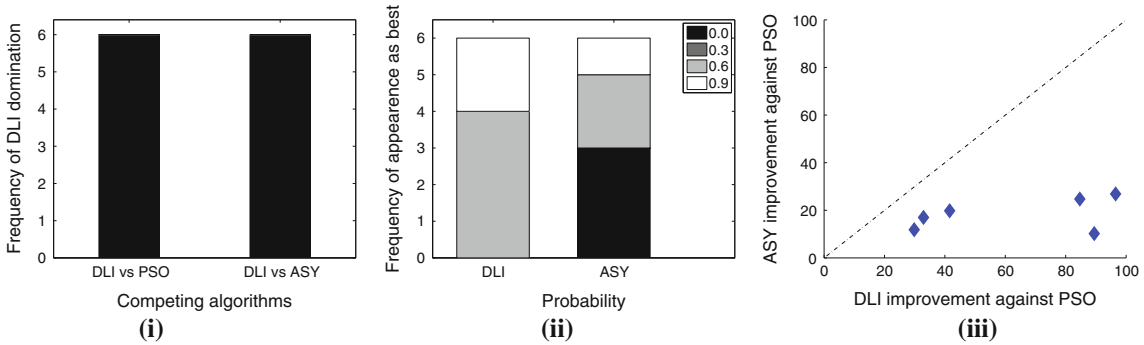


Fig. 5 Frequencies of domination among different algorithm variants and relative performance of PSO-DLI and PSO-ASY with respect to the standard PSO for the nonlinear systems

Table 6 Configuration of the employed GA and DE algorithm

Algorithm	Parameter	Value
GA	Representation	Binary with 32-bit encoding
	Crossover	Single point with probability 0.9
	Mutation	Uniform bit mutation with probability 0.001
	Population update	Replacing 25 % of the population per generation
DE	Variant	rand/1/bin
	F	0.5
	CR	0.7

Table 7 Swarm size and maximum number of function evaluations for the nonlinear systems in Grosan (2008)

Problem	Swarm size	Func. eval.
TP5	500	15×10^4
TP6	300	6×10^4
TP7	500	25×10^4
TP8	500	50×10^4
TP9	500	15×10^4
TP10	500	15×10^4

significantly higher by several orders of magnitude than the one used in our previous experiments. This was beneficial both for the DE approach as well as for DLI0.3, since any possible gain produced by the proposed loss of information scheme was absorbed by the competency of the algorithm due the excessive computational budget. Thus, it can be inferred that increasing the available computational budget produces a tendency of PSO-DLI to approximate the behavior of the standard PSO.

Although the reported results for the rest of the algorithms are rather indicative, PSO-DLI has shown convincing evidence that it can reach high performance standards. This was further verified on harder instances of the standard test suite as reported in the following section.

4.3 Third round of experiments: modified standard test suite

In the third round of experiments, the performance of PSO-DLI was assessed on shifted and rotated versions of the problems of the standard test suite, henceforth called the *shifted test suite*. These problems, denoted as SH-TP0–SH-TP4, were proposed in the IEEE Congress on Evolutionary Computation 2008 (IEEE CEC'08) for large-scale optimization competition (Tang et al. 2007) and they are defined in "Shifted test suite" of Appendix. The PSO-DLI variants used in our previous experiments were applied on the shifted test suite and their performance was compared with that of two PSO-based approaches that participated in the aforementioned competition at IEEE CEC'08, namely the Efficient Population Utilization Strategy for Particle Swarm Optimizer (EPUS-PSO) (Hsieh et al. 2008, 2009) and the Dynamic Multi-Swarm Particle Swarm Optimizer with local search (DMS-PSO) (Zhao et al. 2008).

The test problems were all considered in their 100-dimensional instances, while their ranges were the ones reported in Table 1 for $n = 100$. The parameter configuration of the PSO-DLI variants was identical to that of Table 2 for $n = 100$. This setting is in accordance to the one used for EPUS-PSO and DMS-PSO in Tang et al. (2007) with respect to the available computational budget, i.e., the total number of function evaluations.

The obtained results are reported in Table 9. As required for the specific test problems, the algorithms were compared with respect to the absolute error of the obtained best function value to the true global minimum, averaged over all experiments. The results for EPUS-PSO and DMS-PSO were adopted from their original references (Hsieh et al. 2008; Zhao et al. 2008). In the case of DMS-PSO, the minimum resolution close to zero was not explicitly reported. These cases are marked in Table 9.

As we can see, the PSO-DLI variants produced competitive results. In SH-TP0 and SH-TP3, DLI0.9 strikingly achieved the best performance (excluding the marked

Table 8 Comparative results of PSO-DLI with the MONS, GA and DE algorithm. Results for MONS are reproduced from (Grosan 2008) were no standard errors are reported

		TP5	TP6	TP7	TP8	TP9	TP10
MONS	Mean	1.8000e+00	1.0000e−01	6.0000e−01	1.1000e+00	2.0000e−01	2.0000e−02
	St. err.	–	–	–	–	–	–
GA	Mean	1.0090e−01	1.2900e−02	9.5690e−01	1.0280e+00	4.5340e−01	2.1000e−06
	St. err.	6.2100e−02	2.2900e−02	6.7780e−01	5.5030e−01	4.7370e−01	1.0000e−06
DE	Mean	1.4367e−16	1.2854e−03	1.0077e−02	1.2936e−16	5.2022e−04	6.3720e−03
	St. err.	1.8920e−18	2.4667e−03	9.0729e−04	5.8712e−17	1.9224e−04	3.7393e−03
DLI0.3	Mean	1.4311e−16	6.7117e−11	4.5355e−02	4.9100e−02	2.8222e−03	2.5690e−08
	St. err.	3.9642e−31	6.1695e−10	2.6900e−02	3.7633e−02	2.9754e−03	1.3515e−07
DLI0.6	Mean	1.4311e−16	5.7830e−08	4.7725e−02	6.1449e−02	7.6873e−03	1.9092e−07
	St. err.	3.9642e−31	4.8674e−07	2.9271e−02	5.2835e−02	8.6158e−03	8.7543e−07
DLI0.9	Mean	1.4311e−16	2.4547e−06	5.8572e−02	1.2087e−01	1.8188e−02	1.5300e−06
	St. err.	3.9642e−31	1.8963e−05	3.4466e−02	8.1642e−02	2.0028e−02	4.4812e−06

Table 9 Results for the 100-dimensional instance of the shifted test suite

		SH-TP0	SH-TP1	SH-TP2	SH-TP3	SH-TP4
EPUS-PSO	Mean	7.470e−01	4.990e+03	4.710e+02	3.720e−01	2.060e+00
	St. err.	1.700e−01	5.350e+03	5.940e+01	5.600e−02	4.400e− 01
DMS-PSO	Mean	0.000e+00[a]	2.830e+02	1.829e+02	0.000e+00[a]	0.000e+00[a]
	St. err.	0.000e+00[a]	9.402e+02	2.164e+01	0.000e+00[a]	0.000e+00[a]
DLI0.3	Mean	6.066e−04	9.366e+02	4.649e+02	4.153e−04	1.830e+00
	St. err.	1.425e−04	2.194e+02	4.529e+01	1.459e−04	7.017e−01
DLI0.6	Mean	9.594e−05	7.130e+02	4.634e+02	6.830e−05	2.374e+00
	St. err.	3.038e−05	1.641e+02	4.851e+01	2.144e−05	4.181e−01
DLI0.9	Mean	5.989e−05	7.412e+02	4.566e+02	4.660e−05	2.562e+00
	St. err.	1.789e−05	2.292e+02	5.222e+01	2.570e−05	3.800e−01

The performance of EPUS-PSO and DMS-PSO is reproduced from their original references (Hsieh et al. 2008; Zhao et al. 2008)

[a] It was unclear how close to zero they lie

cases). This is surprising if we take into consideration that PSO-DLI constitutes a very simple approach that does not employ local search or any other sophisticated mechanism specifically suited to this kind of problems, in contrast to the other two approaches. Even for the rest of the problems its performance was highly competitive to the specialized algorithms. This is a very promising evidence that offers motivation for the development of hybrid PSO-DLI variants that can incorporate mechanisms for enhancing their performance in large-scale optimization problems.

4.4 Fourth round of experiments: large-scale competition problems in special issue

Recently, a set of 19 benchmark functions was proposed as a common test suite for the assessment of scalability of metaheuristics on large-scale problems. The test suite constituted the experimental basis for the "*special issue on scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems*" of the Soft Computing journal. General information can be found in the editorial of the special issue (Lozano et al. 2011), while their definitions are freely available on the internet,[3] hence they are omitted due to space limitations. Sixteen algorithms appeared in the special issue and competed against each other on the specific problems for dimensions ranging from 50 up to 1000. The average performance for each algorithm is summarized in a table posted on the previously mentioned web site.[4] The size of the table is prohibitive for replication here, thus we refer the reader directly to the web source.

We considered the specific test suite as our last experimentation framework in the present study, with the 19 test problems being henceforth denoted as SC-TP1–SC-TP19. Since the tuning of PSO-DLI for tackling large-scale

[3] http://sci2s.ugr.es/EAMHCO/testfunctions-SOCO.pdf.

[4] http://sci2s.ugr.es/eamhco/SOCO-results.xls.

Table 10 Average error of PSO-DLI in test problems SC-TP1–SC-TP19

Dim.	SC-TP1	SC-TP2	SC-TP3	SC-TP4	SC-TP5	SC-TP6	SC-TP7	SC-TP8	SC-TP9	SC-TP10
50	0.00e+00	5.20e+00	7.40e+01	0.00e+00	3.10e−02	0.00e+00	0.00e+00	1.12e+03	1.81e+02	1.64e−01
100	0.00e+00	3.47e+01	2.07e+02	0.00e+00	0.00e+00	0.00e+00	1.40e−10	2.10e+04	5.67e+02	1.46e+01
200	0.00e+00	5.65e+01	5.70e+02	0.00e+00	0.00e+00	0.00e+00	1.33e+01	7.92e+04	1.40e+03	3.93e+01
	SC-TP11	SC-TP12	SC-TP13	SC-TP14	SC-TP15	SC-TP16	SC-TP17	SC-TP18	SC-TP19	
50	1.77e+02	3.93e+00	1.62e+02	1.28e+02	0.00e+00	1.27e+01	3.57e+02	7.60e+01	0.00e+00	
100	5.72e+02	3.00e+01	4.18e+02	3.59e+02	0.00e+00	3.48e+02	7.90e+02	2.00e+02	4.89e+00	
200	1.40e+03	3.97e+02	1.39e+03	9.04e+02	1.52e+00	8.96e+02	1.79e+03	4.58e+02	2.50e+01	

problems of very high dimensionality (e.g., 1000) was out of scope of the present study, we restricted our experiments to dimensions 50, 100 and 200, taking a further step from the dimensions considered in the previous section. The configurations of PSO-DLI that were used in the previous sections, were also adopted in this set of experiments.

PSO-DLI was applied on all test problems according to the experimental setup suggested in (Lozano et al. 2011). Its average error was recorded and reported in Table 10. The main purpose was to investigate its ability to approximate the error achieved by the rest of the algorithms, within a prescribed accuracy range. Thus, we compared PSO-DLI's average errors with the corresponding ones reported in the web sources above, for the 16 algorithms that competed in the special issue. The comparisons were made for the different levels of accuracy defined in the following set:

$$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}\}.$$

If PSO-DLI was capable of achieving the same (or better) performance with another algorithm with the selected accuracy level on a specific problem-dimension pair, then this was considered as a successful instance. We recorded the successes of PSO-DLI for all test problems, dimensions, algorithms and accuracy levels, which counts a total of $19 \times 3 \times 16 = 912$ instances per accuracy level. Obviously, the remarkably high accuracy values included above, were considered for revealing the scaling in the successful instances.

The results are graphically illustrated in Fig. 6. Specifically, for each accuracy level there is a bar that shows the number of successful instances (out of 912). Also, each bar is separated in three parts of different colors, denoting the corresponding successful instances per problem dimension. The figure clearly suggests that for demanding accuracies PSO-DLI was capable of achieving the same (or better) performance in more than 1/3 of the cases. This can be consider as a promising sign since PSO-DLI was neither designed nor tuned for such high-dimensional cases as the rest of the algorithms. Nevertheless, it constitutes a strong

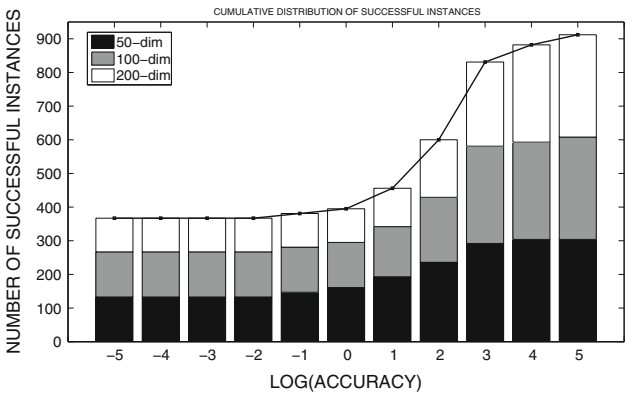


Fig. 6 Cumulative distribution of the number of successful instances per accuracy level

motivation for the future development and improvement of PSO-DLI schemes in order to render it fully competitive to the state-of-the-art in large-scale problems.

As a final observation, we shall mention the consistency of PSO-DLI under dimension and accuracy scaling, as it is implied by the smooth transition of the bars when accuracy increases as well as by the retained proportion of each color in the bars.

5 Conclusions

We introduced PSO-DLI, an algorithm based on the constriction coefficient model of PSO, which promotes asynchronous update and undisrupted movement of the particles accompanied by loss of information. Representative variants of the algorithm were extensively tested both on benchmark problems and challenging applications. Also, they were compared with other widely used algorithms, some of which incorporate sophisticated techniques to tackle specific types of problems.

The obtained results were very promising, offering intuition on PSO-DLI's performance under different experimental settings. The proposed approach was able to outperform the rest of the algorithms in many problem

instances. It was also revealed that in cases where the available computational budget was rather limited, the proposed scheme with higher loss probabilities exhibited very promising performance. On the other hand, in cases where excessive computational budget was available, the gain became higher for smaller loss probabilities indicating that PSO-DLI tends to resemble the standard PSO model.

In addition, PSO-DLI is based on a simple idea that takes full advantage of the dynamics of the constriction coefficient PSO model. Its implementation requires only minor modifications of the original PSO model and alleviates expensive bookkeeping operations. Taking into consideration its encouraging results as well as its asynchronous nature, PSO-DLI could be considered as an effective basis for the development of hybrid PSO-based algorithms for parallel systems and large-scale problems. In fact, this can be a very interesting direction for future research on the proposed algorithm.

Nevertheless, it must be emphasized that PSO-DLI is designed to take full advantage of the exploitation properties that accompany the specific constriction coefficient PSO model. Thus, the generalization of the derived analysis and conclusions in radically different environments such as in integer, noisy or dynamic problems, where the average behavior of the constriction coefficient PSO model is altered, is not straightforward and can constitute a challenging subject for further investigation, along with the demanding large-scale cases.

Acknowledgments The authors wish to thank the anonymous reviewers for their valuable comments and suggestions.

Appendix: Test problems

Standard test suite

The standard test suite consists of the following problems:

Test Problem 0 (TP0-Sphere) (Parsopoulos 2010). This is a separable n-dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n} x_i^2, \tag{8}$$

and it has a single global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 1 (TP1-Generalized Rosenbrock) (Parsopoulos 2010). This is a non-separable n-dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2), \tag{9}$$

and it has a global minimizer, $x^* = (1, 1, \ldots, 1)^\top$, with $f(x^*) = 0$.

Test Problem 2 (TP2-Rastrigin) (Parsopoulos 2010). This is a separable n-dimensional problem, defined as:

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)), \tag{10}$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 3 (TP3-Griewank) (Parsopoulos 2010). This is a non-separable n-dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{11}$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 4 (TP4-Ackley) (Parsopoulos 2010). This is a non-separable n-dimensional problem, defined as:

$$f(x) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right), \tag{12}$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Nonlinear systems

This test set consists of six real-application problems, which are modeled as systems of nonlinear equations. Computing a solution of a nonlinear system is a very challenging task and it has received the ongoing attention of the scientific community. A common methodology for solving such systems is their transformation to an equivalent global optimization problem, which allows the use of a wide range of optimization tools. The transformation produces a single objective function by aggregating all the system's equations, such that the solutions of the original system are exactly the same with that of the derived optimization problem.

Consider the system of nonlinear equations:

$$\begin{cases} f_1(x) = 0, \\ f_2(x) = 0, \\ \quad \vdots \\ f_m(x) = 0, \end{cases}$$

with $x \in S \subset \mathbb{R}^n$. Then, the objective function:

$$f(x) = \sum_{i=1}^{m} |f_i(x)|, \tag{13}$$

defines an equivalent optimization problem. Obviously, if x^* with $f(x^*) = 0$ is a global minimizer of the objective

function, then x^* is also a solution of the corresponding nonlinear system and vice versa.

In our experiments, we considered the following nonlinear systems, previously employed by Grosan and Abraham (2008) to justify the usefulness of evolutionary approaches as efficient solvers of nonlinear systems:

Test Problem 5 (TP5-Interval Arithmetic Benchmark) (Grosan 2008) This problem consists of the following system:

$$\begin{cases} x_1 - 0.25428722 - 0.18324757\,x_4x_3x_9 = 0, \\ x_2 - 0.37842197 - 0.16275449\,x_1x_{10}x_6 = 0, \\ x_3 - 0.27162577 - 0.1695507\,x_1x_2x_{10} = 0, \\ x_4 - 0.19807914 - 0.15585316\,x_7x_1x_6 = 0, \\ x_5 - 0.44166728 - 0.19950920\,x_7x_6x_3 = 0, \\ x_6 - 0.14654113 - 0.18922793\,x_8x_5x_{10} = 0, \\ x_7 - 0.42937161 - 0.21180486\,x_2x_5x_8 = 0, \\ x_8 - 0.07056438 - 0.17081208\,x_1x_7x_6 = 0, \\ x_9 - 0.34504906 - 0.19612740\,x_{10}x_6x_8 = 0, \\ x_{10} - 0.42651102 - 0.21466544\,x_4x_8x_1 = 0. \end{cases} \tag{14}$$

The resulting objective function defined by Eq. (13), is 10-dimensional with global minimum $f(x^*) = 0$.

Test Problem 6 (TP6-Neurophysiology Application) (Grosan 2008) This problem consists of the following system:

$$\begin{cases} x_1^2 + x_3^2 = 1, \\ x_2^2 + x_4^2 = 1, \\ x_5x_3^3 + x_6x_4^3 = c_1, \\ x_5x_1^3 + x_6x_2^3 = c_2, \\ x_5x_1x_3^2 + x_6x_4^2x_2 = c_3, \\ x_5x_1^2x_3 + x_6x_2^2x_4 = c_4, \end{cases} \tag{15}$$

where the constants, $c_i = 0$, $i = 1, 2, 3, 4$. The resulting objective function is six-dimensional with global minimum $f(x^*) = 0$.

Test Problem 7 (TP7-Chemical Equilibrium Application) (Grosan 2008) This problem consists of the following system:

$$\begin{cases} x_1x_2 + x_1 - 3x_5 = 0, \\ 2x_1x_2 + x_1 + x_2x_3^2 + R_8x_2 - Rx_5 + 2R_{10}x_2^2 + R_7x_2x_3 \\ +R9x_2x_4 = 0, \\ 2x_2x_3^2 + 2R_5x_3^2 - 8x_5 + R_6x_3 + R_7x_2x_3 = 0, \\ R_9x_2x_4 + 2x_4^2 - 4Rx_5 = 0, \\ x_1(x_2 + 1) + R_{10}x_2^2 + x_2x_3^2 + R_8x_2 + R_5x_3^2 \\ +x_4^2 - 1 + R_6x_3 + R_7x_2x_3 + R_9x_2x_4 = 0, \end{cases} \tag{16}$$

where,

$$R = 10, \quad R_5 = 0.193, \quad R_6 = \frac{0.002597}{\sqrt{40}}, \quad R_7 = \frac{0.003448}{\sqrt{40}},$$

$$R_8 = \frac{0.00001799}{40}, \quad R_9 = \frac{0.0002155}{\sqrt{40}}, \quad R_{10} = \frac{0.00003846}{40}.$$

The corresponding objective function is five-dimensional with global minimum $f(x^*) = 0$.

Test Problem 8 (TP8-Kinematic Application) (Grosan 2008). This problem consists of the following system:

$$\begin{cases} x_i^2 + x_{i+1}^2 - 1 = 0, \\ a_{1i}x_1x_3 + a_{2i}x_1x_4 + a_{3i}x_2x_3 + a_{4i}x_2x_4 + a_{5i}x_2x_7 + \\ a_{6i}x_5x_8 + a_{7i}x_6x_7 + a_{8i}x_6x_8 + a_{9i}x_1 + a_{10i}x_2 + a_{11i}x_3 + \\ a_{12i}x_4 + a_{13i}x_5 + a_{14i}x_6 + a_{15i}x_7 + a_{16i}x_8 + a_{17i} = 0, \end{cases} \tag{17}$$

with a_{ki}, $1 \le k \le 17$, $1 \le i \le 4$, is the corresponding element of the kth row and ith column of the matrix:

$$A = \begin{bmatrix} -0.249150680 & 0.125016350 & -0.635550077 & 1.48947730 \\ 1.609135400 & -0.686607360 & -0.115719920 & 0.23062341 \\ 0.279423430 & -0.119228120 & -0.666404480 & 1.32810730 \\ 1.434801600 & -0.719940470 & 0.110362110 & -0.25864503 \\ 0.000000000 & -0.432419270 & 0.290702030 & 1.16517200 \\ 0.400263840 & 0.000000000 & 1.258776700 & -0.26908494 \\ -0.800527680 & 0.000000000 & -0.629388360 & 0.53816987 \\ 0.000000000 & -0.864838550 & 0.581404060 & 0.58258598 \\ 0.074052388 & -0.037157270 & 0.195946620 & -0.20816985 \\ -0.083050031 & 0.035436896 & -1.228034200 & 2.68683200 \\ -0.386159610 & 0.085383482 & 0.000000000 & -0.69910317 \\ -0.755266030 & 0.000000000 & -0.079034221 & 0.35744413 \\ 0.504201680 & -0.039251967 & 0.026387877 & 1.24991170 \\ -1.091628700 & 0.000000000 & -0.057131430 & 1.46773600 \\ 0.000000000 & -0.432419270 & -1.162808100 & 1.16517200 \\ 0.049207290 & 0.000000000 & 1.258776700 & 1.07633970 \\ 0.049207290 & 0.013873010 & 2.162575000 & -0.69686809 \end{bmatrix}.$$

The corresponding objective function is eight-dimensional with global minimum $f(x^*) = 0$.

Test Problem 9 (TP9-Combustion Application) (Grosan 2008). This problem consists of the following system:

$$\begin{cases} x_2 + 2x_6 + x_9 + 2x_{10} = 10^{-5}, \\ x_3 + x_8 = 3 \times 10^{-5}, \\ x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} = 5 \times 10^{-5}, \\ x_4 + 2x_7 = 10^{-5}, \\ 0.5140437 \times 10^{-7}x_5 = x_1^2, \\ 0.1006932 \times 10^{-6}x_6 = 2x_2^2, \\ 0.7816278 \times 10^{-15}x_7 = x_4^2, \\ 0.1496236 \times 10^{-6}x_8 = x_1x_3, \\ 0.6194411 \times 10^{-7}x_9 = x_1x_2, \\ 0.2089296 \times 10^{-14}x_{10} = x_1x_2^2. \end{cases} \tag{18}$$

The corresponding objective function is ten-dimensional with global minimum $f(x^*) = 0$.

Test Problem 10 (TP10-Economics Modeling Application) (Grosan 2008) This problem consists of the following system:

$$\begin{cases} \left(x_k + \sum_{i=1}^{n-k-1} x_ix_{i+k}\right)x_n - c_k = 0, \\ \sum_{l=1}^{n-1} x_l + 1 = 0, \end{cases} \tag{19}$$

where $1 \leq k \leq n-1$, and $c_i = 0, i = 1, 2, \ldots, n$. The problem was considered in its 20-dimensional instance. Thus, the corresponding objective function was also 20-dimensional, with global minimum $f(x^*) = 0$.

Shifted test suite

The problems in the shifted test suite are defined as follows:

Shifted Test Problem 0 (SH-TP0-Sphere) (Tang et al. 2007). This is a separable n-dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n} (x_i - o_i)^2 + f_{\text{bias}}, \tag{20}$$

and it has one global minimizer, $x^* = (o_1, o_2, \ldots, o_n)^\top$, with $f(x^*) = f_{\text{bias}} = -450$.

Shifted Test Problem 1 (SH-TP1-Generalized Rosenbrock) (Tang et al. 2007) This is a non-separable n-dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n-1} \left(100\left(z_{i+1} - z_i^2\right)^2 + (z_i - 1)^2\right) + f_{\text{bias}}, \tag{21}$$

where $z = x - o + (1, 1, \ldots, 1)^\top$, and it has a global minimizer, $x^* = (o_1, o_2, \ldots, o_n)^\top$, with $f(x^*) = f_{\text{bias}} = 390$.

Shifted Test Problem 2 (SH-TP2-Rastrigin) (Tang et al. 2007). This is a separable n-dimensional problem, defined as:

$$f(x) = 10n + \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i)) + f_{\text{bias}}, \tag{22}$$

where $z = x - o$, and it has a global minimizer, $x^* = (o_1, o_2, \ldots, o_n)^\top$, with $f(x^*) = f_{\text{bias}} = -330$.

Shifted Test Problem 3 (SH-TP3-Griewank) (Tang et al. 2007). This is a non-separable n-dimensional problem, defined as:

$$f(x) = \sum_{i=1}^{n} \frac{z_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{\text{bias}}, \tag{23}$$

where $z = x - o$, and it has a global minimizer, $x^* = (o_1, o_2, \ldots, o_n)^\top$, with $f(x^*) = f_{\text{bias}} = -180$.

Shifted Test Problem 4 (SH-TP4-Ackley) (Tang et al. 2007). This is a non-separable n-dimensional problem, defined as:

$$f(x) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi z_i)\right) + f_{\text{bias}}, \tag{24}$$

where $z = x - o$, and it has a global minimizer, $x^* = (o_1, o_2, \ldots, o_n)^\top$, with $f(x^*) = f_{\text{bias}} = -140$.

References

Akat SB, Gazi V (2008) Decentralized asynchronous particle swarm optimization. In: Proceedings of the IEEE 2008 swarm intelligence symposium, pp 1–8

Bäck T, Fogel D, Michalewicz Z (1997) Handbook of evolutionary computation. IOP Publishing and Oxford University Press, New York

Bartz-Beielstein T, Parsopoulos KE, Vrahatis MN (2004) Design and analysis of optimization algorithms using computational statistics. Appl Numer Anal Comput Math 1(2):413–433

Blackwell T, Branke J (2006) Multi-swarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans Evolut Comput 10(4):459–472

Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73

Desell T, Magdon-Ismail M, Szymanski B, Varela C, Newberg H, Cole N (2009) Robust asynchronous optimization for volunteer computing grids. In: Proceedings of the 5th IEEE international conference on e-Science, pp 263–270

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings sixth symposium on micro machine and human science, pp 39–43, IEEE Service Center, Piscataway, NJ, 1995

Gazi V (2007) Asynchronous particle swarm optimization. In: Proceedings of the IEEE 15th conference on signal processing and communications applications, pp 1–4

Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison Wesley, Reading

Grosan C, Abraham A (2008) A new approach for solving nonlinear equations systems. IEEE Trans Syst Man Cybern Part A: Syst Hum 38(3):698–714

Hernane S, Hernane Y, Benyettou M (2010) An asynchronous parallel particle swarm optimization algorithm for a scheduling problem. J Appl Sci 10(8):664–669

Hsieh S-T, Sun T-Y, Liu C-C, Tsai S-J (2008) Solving large scale global optimization using improved particle swarm optimizer. In: Proceedings of the IEEE 2008 congress on evolutionary computation, Hong Kong, pp 1777–1784

Hsieh S-T, Sun T-Y, Liu C-C, Tsai S-J (2009) Efficient population utilization strategy for particle swarm optimizer. IEEE Trans Syst Man Cybern Part B: Cybern 39(2):444–456

Kennedy J (1999) Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of IEEE congress evolutionary computation, pp 1931–1938. IEEE Press, Washington

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference neural networks, vol IV, pp 1942–1948. IEEE Service Center, Piscataway

Koh B-I, George AD, Haftka RT, Fregly BJ (2006) Parallel asynchronous particle swarm optimization. Int J Numer Methods Eng 67:578–595

Lozano M, Molina D, Herrera F (2011) Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft Comput 15(11):2085–2087

Parsopoulos KE, Vrahatis MN (2002a) Initializing the particle swarm optimizer using the nonlinear simplex method. In: Grmela A, Mastorakis NE (eds) Advances in intelligent systems, fuzzy systems, evolutionary computation. WSEAS Press, pp 216–221

Parsopoulos KE, Vrahatis MN (2002b) Recent approaches to global optimization problems through particle swarm optimization. Nat Comput 1(2–3):235–306

Parsopoulos KE, Vrahatis MN (2010) Particle swarm optimization and intelligence: advances and applications. Information Science Publishing (IGI Global), Hershey, PA

Poli R (2007) An analysis of publications on particle swarm optimisation applications. Technical Report CSM-649, University of Essex, Department of Computer Science, UK

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359

Suganthan PN (1999) Particle swarm optimizer with neighborhood operator. In: Proceedings of IEEE congress evolutionary computation, pp 1958–1961, Washington, USA

Tang K, Yao X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (2007) Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China, China

Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85:317–325

Zhao SZ, Liang JJ, Suganthan PN, Tasgetiren MF (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: Proceedings of the IEEE 2008 congress on evolutionary computation, Hong Kong, pp 3845–3852